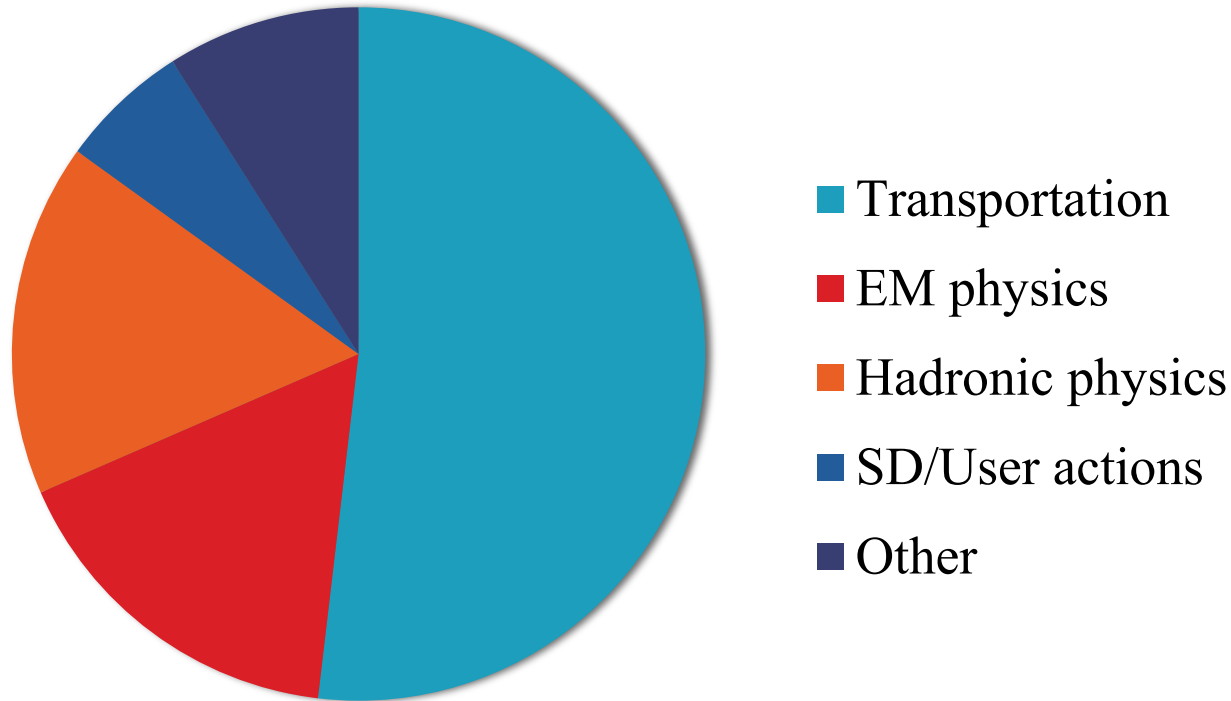


Geant4 status in CMS

- Production version of Geant4 for 2015
 - Geant4 version 10.0p02 built in sequential mode
 - Production platform slc6_amd64_gcc491
 - Default physics list: **QGSP_FTFP_BERT_EML**
 - ~8 billion events already produced in 2015
 - This number will increase for the end of the year
- Current development versions of Geant4 in CMSSW
 - Geant4 10.0p03 + patch of Geant4e for threading is established
 - Multi-threaded Geant4 is fully integrated with CMS multi-threaded framework
 - Our goal is to use it in production 2016
 - Platform slc6_amd64_gcc493
 - Geant4 10.1p02 is also available in development branches
 - Geant4 10.2 planned to be our production version in 2017

Where our simulation CPU goes (CHEP'15 talk)



- Technical performance improvements for Run 2 simulation:
 1. Upgrade to Geant4 10.0 (~5%)
 2. Implementation of Russian Roulette technique (~30%)
 3. CMSSW code optimization (~15%)
 4. Library repackaging (~10%)

10% performance gain from hidden visibility without playing with linker scripts (CHEP'15 talk)

Repackage all shared libraries in CMSSW that depend on Geant4 into a single static library to “hide” Geant4 from the rest of CMSSW

- Use single archive library for Geant4 itself
- This allowed us to more aggressively optimize at link time: adding “**-flto -Wl,--exclude-libs,ALL**” works best

Constraints this imposes:

- Must control dependencies to use Geant4 only within this single library:
 - This is “easy” for simulation (<2% of our libraries)
 - However, extending this idea to something effecting the full reconstruction is difficult
- Impact on simulation code developers minimized by keeping .so cached in release. Static library rebuild is the only extra step if developer builds a package in this static library.

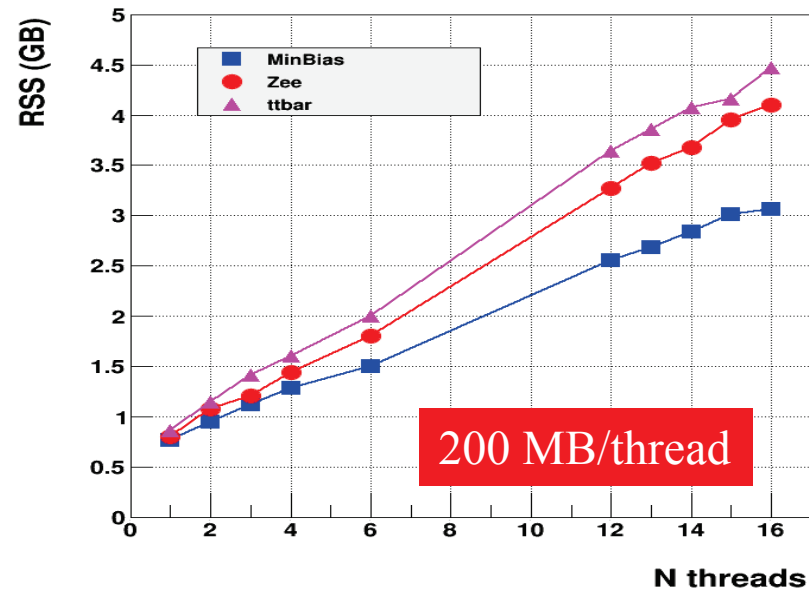
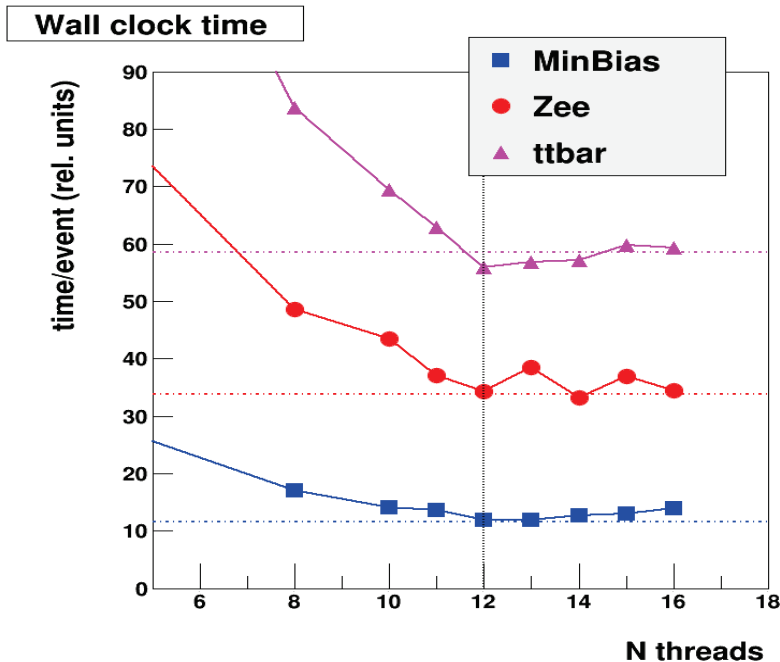
Current performance of CMS



MT GEN-SIM (CHEP'15 talk)

Time/event decreases until the # of threads is equal to the # of cores

Example memory savings: a single 12 threaded MT job requires **~4 GB** RSS instead of **11 GB** for 12 single threaded jobs

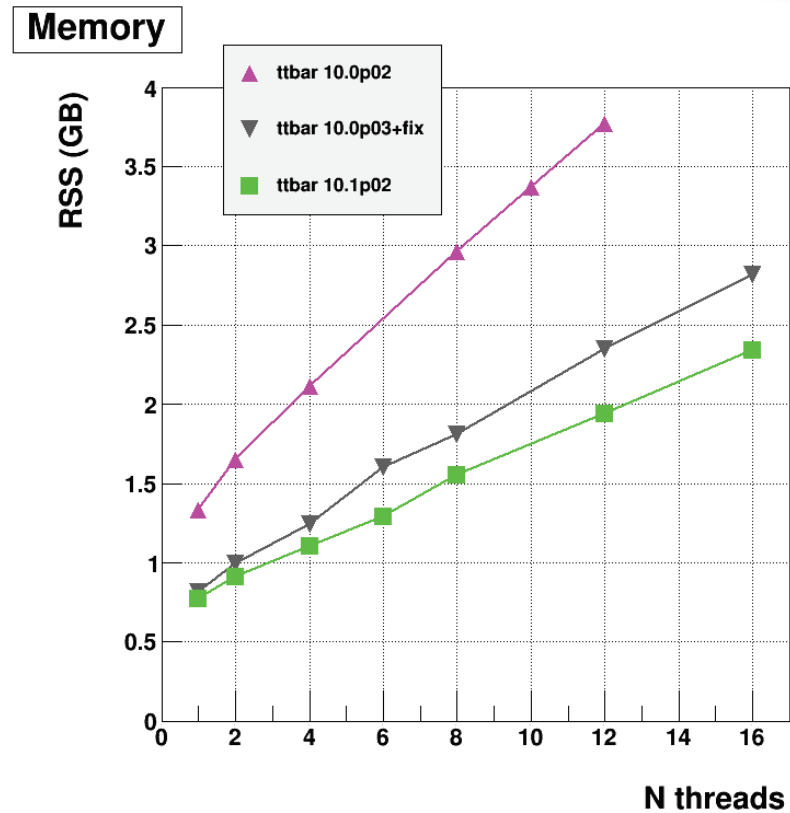


----- Extrapolation
from single core

Excellent scaling performance seen in our tests so far.
Geant4 version 10.0p02

Performance results for Geant4 10.1p02

- **For CMS CPU performance is the same in 10.1 and 10.0**
- Memory grows was observed in 10.0p02 about 1.3 M/event
- There is two main contributions to the memory grows:
 - Memory leak in gamma-nuclear and FTF models
 - Objects created with G4Allocator were not deleted
 - Ineffective data structure for nuclear gamma evaporation models
 - Significant memory was used per thread for nuclear level data
- Both problems were fixed in 10.1p02 and these fixes were backported to CMSSW on top of 10.0p03
- **After backport of fixes required RSS memory for 10.0p03+fix become very similar to 10.1p02**



For 10.0p02 results are shown after 1000 events